

examples11/

```
1: README
2:
3: Computer Science E-259
4:
5:
6: OVERVIEW
7:
8: In these directories are two client-server pairs, one that
9: demonstrates Project 4's warehouse and one that
10: demonstrates a tax service. Each's usage is documented below.
11:
12:
13:
14: TAXES
15:
16: To access TaxService.jws as a web service, simply
17: do the following.
18:
19: 1.  Spawn Tomcat from within examples11/server/, after
20:     specifying in examples11/server/conf/server.xml the
21:     ports desired.
22:
23: 2.  Execute the following command (which, unfortunately, wraps onto
24:     a second line) from within examples11/clients/taxes/.
25:
26:     java org.apache.axis.wsdl.WSDL2Java
27:     "http://$ICEBOX.fas.harvard.edu:n/taxes/TaxService.jws?wsdl"
28:
29: 3.  Edit the first line of
30:     examples11/clients/taxes/TaxClient.java so that
31:     the appropriately named package is imported. The first line of
32:     that file, at present, is
33:
34:     import edu.harvard.fas.icel.taxes.TaxService_jws.*;
35:
36:     because we whipped up this demo on icel.fas.harvard.edu.
37:
38: 4.  Execute the following commands from within
39:     examples11/clients/taxes/.
40:
41:     javac TaxClient.java
42:     java TaxClient
43:
44:     You should see the result of the service's performing the
45:     various calculations specified in TaxClient.java.
46:
47:
48: WAREHOUSE
49:
50: To access Project 4's warehouse as a web service, simply
51: do the following.
52:
53: 1.  Spawn Tomcat from within examples11/server/, after
54:     specifying in examples11/server/conf/server.xml the
55:     ports desired.
56:
57: 2.  In another terminal, execute the following command (which,
58:     unfortunately, wraps onto a second line) from within
59:     examples11/clients/warehouse/.
60:
61:     java org.apache.axis.wsdl.WSDL2Java
```

```
62:      "http://$ICEBOX.fas.harvard.edu:n/warehouse/services/Purchasing?wsdl"
63:
64: 3.  Edit the first line of
65:      examples11/clients/warehouse/PurchasingClient.java so that
66:      the appropriately named package is imported.  The first line of
67:      that file, at present, is
68:
69:      import edu.harvard.fas.icel.warehouse.services.Purchasing.*;
70:
71:      because we whipped up this demo on icel.fas.harvard.edu.
72:
73: 4.  Execute the following commands from within
74:      examples11/clients/warehouse/.
75:
76:      javac PurchasingClient.java
77:      java PurchasingClient
78:
79:      You should see the result of the service's application of po-ack.xsl
80:      to the empty <PO/> element passed to the service by PurchasingClient.
81:
```

```
1: import edu.harvard.fas.icel.taxes.TaxService_jws.*;
2: import org.apache.axis.AxisFault;
3:
4:
5: /**
6:  * Lecture 11's demo of a client that accesses a web service
7:  * that performs various tax calculations.
8:  *
9:  * Excerpted from
10:  * http://www.ammai.com/modules.php?op=modload&name=Sections&file=index&req=viewarticle&artid=4&page=5
11:  *
12:  * @author Ammai.com
13:  * @author Computer Science E-259
14:  */
15:
16: public class TaxClient
17: {
18:     /**
19:      * Main driver.
20:      */
21:     public static void main(String[] args)
22:     {
23:         try
24:         {
25:             // Make a service
26:             TaxServiceService service = new TaxServiceServiceLocator();
27:             TaxService port = service.getTaxService();
28:
29:             // Make the actual calls to the three methods
30:             double taxpercent = port.calcTaxRate(21.00, 23.10);
31:             double total = port.calcTotal(21.00, 0.10);
32:             double subtotal = port.calcSubTotal(23.10, 0.10);
33:
34:             // Output the results
35:             System.out.println(
36:                 "Subtotal: 21.00, Total: 23.10, Tax: " + taxpercent
37:             );
38:             System.out.println(
39:                 "Subtotal: 21.00, Tax: 0.10, Total: " + total
40:             );
41:             System.out.println(
42:                 "Total: 23.10, Tax: 0.10, Subtotal: " + subtotal
43:             );
44:         }
45:         catch (AxisFault af)
46:         {
47:             System.err.println("An Axis Fault occurred: " + af);
48:         }
49:         catch (Exception e)
50:         {
51:             System.err.println("Exception caught: " + e);
52:         }
53:     }
54: }
```

```
1: import edu.harvard.fas.icel.warehouse.services.Purchasing.*;
2: import org.apache.axis.AxisFault;
3:
4:
5: /**
6:  * Lecture 11's demo of a client that accesses Project 4's
7:  * warehouse.
8:  *
9:  * @author Computer Science E-259
10:  */
11:
12: public class PurchasingClient
13: {
14:     /**
15:      * Main driver.
16:      */
17:     public static void main(String [] argv)
18:     {
19:         // attempt to connect to warehouse
20:         try
21:         {
22:             PurchasingService service = new PurchasingServiceLocator();
23:             Purchasing port = service.getPurchasing();
24:
25:             // eh, we've got nothing to say tonight
26:             System.out.println(port.processPO("<PO/>"));
27:         }
28:         catch (AxisFault af)
29:         {
30:             System.err.println("An Axis Fault occurred: " + af);
31:             System.err.println();
32:             System.err.println(af.dumpToString());
33:         }
34:         catch (Exception e)
35:         {
36:             System.err.println("Exception caught: " + e);
37:         }
38:     }
39: }
```

```
1: /**
2:  * Lecture 11's demo of a web service that performs
3:  * various tax calculations.
4:  *
5:  * Excerpted from
6:  * http://www.ammai.com/modules.php?op=modload&name=Sections&file=index&req=viewarticle
&artid=4&page=4.
7:  *
8:  * @author Ammai.com
9:  * @author Computer Science E-259
10: **/
11:
12: public class TaxService
13: {
14:     /**
15:      * Determines tax rate based on given subtotal and total.
16:      */
17:     public double calcTaxRate(double subtotal, double total)
18:     {
19:         double rate = (total - subtotal) / subtotal;
20:         return rate;
21:     }
22:
23:
24:     /**
25:      * Determines pre-tax subtotal based on given total and
26:      * tax rate.
27:      */
28:     public double calcSubTotal(double total, double taxpercent)
29:     {
30:         double subtotal = total / (1 + taxpercent);
31:         return subtotal;
32:     }
33:
34:
35:     /**
36:      * Determines total based on given subtotal and
37:      * tax rate.
38:      */
39:     public double calcTotal(double subtotal, double taxpercent)
40:     {
41:         double total = subtotal * (1 + taxpercent);
42:         return total;
43:     }
44: }
```